The Impact of Correlated Metrics on the Interpretation of Defect Models

Jirayus Jiarpakdee, *Student Member, IEEE,* Chakkrit Tantithamthavorn, *Member, IEEE,* and Ahmed E. Hassan, *Fellow, IEEE*

Abstract—Defect models are analytical models for building empirical theories related to software quality. Prior studies often derive knowledge from such models using interpretation techniques, e.g., ANOVA Type-I. Recent work raises concerns that correlated metrics may impact the interpretation of defect models. Yet, the impact of correlated metrics in such models has not been investigated. In this paper, we investigate the impact of correlated metrics on the interpretation of defect models and the improvement of the interpretation of defect models when removing correlated metrics. Through a case study of 14 publicly- available defect datasets, we find that (1) correlated metrics have the largest impact on the consistency, the level of discrepancy, and the direction of the ranking of metrics, especially for ANOVA techniques. On the other hand, we find that removing all correlated metrics (2) improves the consistency of the produced rankings regardless of the ordering of metrics (except for ANOVA Type-I); (3) improves the consistency of ranking of metrics among the studied interpretation techniques; (4) impacts the model performance by less than 5 percentage points. Thus, when one wishes to derive sound interpretation from defect models, one must (1) mitigate correlated metrics especially for ANOVA analyses; and (2) avoid using ANOVA Type-I even if all correlated metrics are removed.

Index Terms—Software Quality Assurance, Defect models, Hypothesis Testing, Correlated Metrics, Model Specification.

1 Introduction

Defect models are constructed using historical software project data to identify defective modules and explore the impact of various phenomena (i.e., software metrics) on software quality. The interpretation of such models is used to build empirical theories that are related to software quality (i.e., what software metrics share the strongest association with software quality?). These empirical theories are essential for project managers to chart software quality improvement plans to mitigate the risk of introducing defects in future releases (e.g., a policy to maintain code as simple as possible).

Plenty of prior studies investigate the impact of many phenomena on code quality using software metrics, for example, code size, code complexity [31, 49, 71], change complexity [42, 57, 59, 71, 88], antipatterns [41], developer activity [71], developer experience [61], developer expertise [5], developer and reviewer knowledge [81], design [3, 10, 11, 14, 16], reviewer participation [50, 82], code smells [40], and mutation testing [7]. To perform such studies, there are five common steps: (1) formulating of hypotheses that pertain to the phenomena that one wishes to study; (2) designing appropriate metrics to operationalize the intention behind the phenomena under study; (3) defining a model specification (e.g., the ordering of metrics) to be used when constructing an analytical model; (4) constructing an analytical model using, for example, regression models [5, 57, 81, 82, 87] or random forest models [23, 38, 55, 64]; and (5) examining the ranking of metrics using a model interpretation technique (e.g., ANOVA Type-I, one of the most commonly-used interpretation techniques since it is the default built-in function for logistic regression (`glm`) models in R) in order to test the hypotheses.

For example, to study whether complex code increases project risk, one might use the number of reported bugs (`bugs`) to capture risk, and the McCabe's cyclomatic complexity (`CC`) to capture code complexity, while controlling for code size (`size`). We note that one needs to use control metrics to ensure that findings are not due to confounding factors (e.g., large modules are more likely to have more bugs). Then, one must construct an analytical model with a model specification of `bugs`∼`size+CC`. One would then use an interpretation technique (e.g. ANOVA Type-I) to determine the ranking of metrics (i.e., which metrics have a strong relationship with bugs).

Metrics of prior studies are often correlated [22, 32, 33, 35, 36, 74, 77, 85]. For example, Herraiz *et al.* [33], and Gil *et al.* [22] point out that code complexity (`CC`) is often correlated with code size (`size`). Zhang *et al.* [85] point out that many metric aggregation schemes (e.g., averaging or summing of McCabe's cyclomatic complexity values at the function level to derive file-level metrics) often produce correlated metrics.

Recent studies raise concerns that correlated metrics may impact the interpretation of defect models [77, 85]. Our preliminary analysis (PA2) also shows that simply rearranging the ordering of correlated metrics in the model specification (e.g., from `bugs`∼`size+CC` to `bugs`∼`CC+size`) would lead to a different ranking of metrics—i.e., the importance scores are sensitive to the ordering of correlated metrics in a model specification. Thus, if one wants to show that code complexity is strongly associated with risk in a project,

- *J. Jiarpakdee and C. Tantithamthavorn are with the Faculty of Information Technology, Monash University, Australia. E-mail: jirayus.jiar@gmail.com, chakkrit.tantithamthavorn@monash.edu.*
- *A. E. Hassan is with the School of Computing, Queen's University, Canada. E-mail: ahmed@cs.queensu.ca.*

one simply needs to put code complexity (`CC`) as the first metric in their models (i.e., `bugs~CC+size`), even though a more careful analysis would show that `CC` is not associated with `bugs` at all. The sensitivity of the model specification when correlated metrics are included in a model is a critical problem, since the contribution of many prior studies can be altered by simply re-ordering metrics in the model specification if correlated metrics are not properly mitigated. Unfortunately, a literature survey of Shihab [67] shows that as much as 63% of defect studies that are published during 2000-2011 do not mitigate correlated metrics prior to constructing defect models.

In this paper, we set out to investigate (1) the impact of correlated metrics on the interpretation of defect models. After removing correlated metrics, we investigate (2) the consistency of the interpretation of defect models; and (3) its impact on the performance and stability of defect models. In order to detect and remove correlated metrics, we apply the variable clustering (VarClus) and the variance inflation factor (VIF) techniques. We construct logistic regression and random forest models using mitigated (i.e., no correlated metrics) and non-mitigated datasets (i.e., not treated). Finally, we apply 9 model interpretation techniques, i.e., ANOVA Type-I, 4 test statistics of ANOVA Type-II (i.e., Wald, Likelihood Ratio, F, and Chi-square), scaled and non-scaled Gini Importance, and scaled and non-scaled Permutation Importance. We then compare the performance and interpretation of defect models that are constructed using mitigated and non-mitigated datasets. Through a case study of 14 publicly-available defect datasets of systems that span both proprietary and open source domains, we address the following four research questions:

(RQ1) How do correlated metrics impact the interpretation of defect models?
ANOVA Type-I and Type-II often produce the lowest consistency and the highest level of discrepancy of the top-ranked metric, and have the highest impact on the direction of the ranking of metrics between mitigated and non-mitigated models when compared to Gini and Permutation Importance. This finding highlights the risks of not mitigating correlated metrics in the ANOVA analyses of prior studies.

(RQ2) After removing all correlated metrics, how consistent is the interpretation of defect models among different model specifications?
After removing all correlated metrics, the top-ranked metric according to ANOVA Type-II, Gini Importance, and Permutation Importance are consistent. However, the top-ranked metric according to ANOVA Type-I is inconsistent, since the ranking of metrics is impacted by its order in the model specification when analyzed using ANOVA Type-I (which is the default analysis for the `glm` model in R and is commonly-used in prior studies). This finding suggests that ANOVA Type-I must be avoided even if all correlated metrics are removed.

(RQ3) After removing all correlated metrics, how consistent is the interpretation of defect models among the studied interpretation techniques?
After removing all correlated metrics, we find that the consistency of the ranking of metrics among the studied interpretation techniques is improved by 15%- 64% for the top-ranked metric and 21%-71% for the top-3 ranked metrics, respectively, highlighting the benefits of removing all correlated metrics on the interpretation of defect models, i.e., the conclusions of studies that rely on one interpretation technique may not pose a threat after mitigating correlated metrics.

(RQ4) Does removing all correlated metrics impact the performance and stability of defect models?
Removing all correlated metrics impacts the AUC, F-measure, and MCC performance of defect models by less than 5 percentage points, suggesting that researchers and practitioners should remove correlated metrics with care especially for safety-critical software domains.

Based on our findings, we suggest that: *When the goal is to derive sound interpretation from defect models, our results suggest that future studies must (1) mitigate correlated metrics prior to constructing a defect model, especially for ANOVA analyses; and (2) avoid using ANOVA Type-I even if all correlated metrics are removed, but instead opt to use ANOVA Type-II and Type-III for additive and interaction models, respectively. Due to the variety of the built-in interpretation techniques and their settings, our paper highlights the essential need for future studies to report the exact specification (i.e., model formula) of their models and settings (e.g., the calculation methods of the importance score) of the used interpretation techniques.*

1.1 Novelty Statements

To the best of our knowledge, this paper is the first to present:

(1) A series of preliminary analyses of the nature of correlated metrics in defect datasets and their impact on the interpretation of defect models (Appendix 2).
(2) An investigation of the impact of correlated metrics on the consistency, the level of discrepancy, and the direction of the produced rankings by the interpretation techniques (RQ1).
(3) An empirical evaluation of the consistency of such rankings after removing all correlated metrics (RQ2, RQ3).
(4) An investigation of the impact of removing all correlated metrics on the performance and stability of defect models (RQ4).

1.2 Paper Organization

Section 2 discusses the analytical modelling process, correlated metrics and concerns in the literature, and techniques for mitigating correlated metrics. Section 3 describes the design of our case study, while Section 4 presents our results with respect to our four research questions. Section 5 provides practical guidelines for future studies. Section 6 discusses the threats to the validity of our study. Section 7 draws conclusions. For the detailed explanation of the studied correlation analysis techniques, commonly-used analytical learners, and interpretation techniques, see Appendix 1 of the supplementary materials.

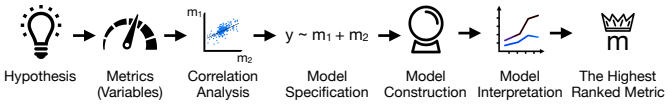


Figure 1: An overview of the analytical modelling process.

2 Background and Motivation

2.1 Analytical Modelling Process

Figure 1 provides an overview of the commonly-used analytical modelling process. First, one must formulate a set of hypotheses pertaining to phenomena of interest (e.g., whether the size of a module increases the risk associated with that module). Second, one must determine a set of metrics which operationalize the hypothesis of interest (e.g., the total lines of code for size, and the number of field reported bugs to capture the risk that is associated with a module). Third, one must perform a correlation analysis to remove correlated metrics. Forth, one must define a model specification (e.g., the ordering of metrics) to be used when constructing an analytical model. Fifth, one is then ready to construct an analytical model using a machine learning technique (e.g., a random forest model) or a statistical learning technique (e.g., a regression model). Finally, one analyzes the ranking of the metrics using model interpretation techniques (e.g., ANOVA or Breiman's Variable Importance) in order to test the hypotheses of interest. The importance ranking of the metrics is essential for project managers to chart appropriate software quality improvement plans to mitigate the risk of introducing defects in future releases. For example, if code complexity is identified as the top-ranked metric, project managers then can suggest developers to reduce the complexity of their code to reduce the risk of introducing defects.

2.2 Correlated Metrics and Concerns in the Literature

Correlated metrics are metrics (i.e., independent variables) that share a strong linear correlation among themselves. In this paper, we focus on two types of correlation among metrics, i.e., collinearity and multicollinearity. Collinearity is a phenomenon in which one metric can be linearly predicted by another metric. On the other hand, multicollinearity is a phenomenon in which one metric can be linearly predicted by a combination of two or more metrics.

Prior work points out that software metrics are often correlated [22, 32, 33, 35, 36, 74, 77, 85]. However, little is known the prevalence of correlated metrics in the publicly-available defect datasets. Thus, we set out to investigate how many defect datasets of which metrics that share a strong relationship with defect-proneness are correlated. Unfortunately, the results of our preliminary analysis (PA1) show that correlated metrics that share a strong relationship with defect-proneness are prevalent in 83 of the 101 (82%) publicly avaliable defect datasets.

In addition, prior work raises concerns that correlated metrics may impact the interpretation of defect models [77, 85]. To better understand how correlated metrics impact the interpretation of defect models, we set out to investigate (1) the impact of the number of correlated metrics on the importance scores of metrics, and (2) the impact of the ordering of correlated metrics in a model specification on the importance ranking metrics. The results of our preliminary analyses (PA2, and PA3) show that the importance scores of metrics substantially decrease when there are correlated metrics in the models for both ANOVA analyses of logistic regression and Variable Importance analyses (i.e., Gini and Permutation) of random forest. The importance scores of metrics are also sensitive to the ordering of correlated metrics (except for ANOVA Type-II).

2.3 Techniques for Mitigating Correlated Metrics

There is a plethora of techniques that have been used to mitigate irrelevant and correlated metrics in the domain of defect prediction, e.g., dimensionality reduction [47, 57, 85], feature selection [1, 54], and correlation analysis [15, 68, 82].

Dimensionality reduction transforms an initial set of metrics into a set of transformed metrics that is representative to the initial set of metrics. Prior work has adopted dimensionality reduction techniques (e.g., Principal Component Analysis) to mitigate correlated metrics and improve the performance of defect models [47, 57, 85]. Since the set of transformed metrics does not hold the assumption of the initial set of metrics, and is not sensible for model interpretation and statistical inference [74], we exclude dimensionality reduction techniques from this paper.

Feature selection produces an optimal subset of metrics that are relevant and non-correlated. One of the most commonly-used feature selection techniques is the correlation-based feature selection technique (CFS) [25] which searches for the best subset of metrics that share the highest correlation with the outcome (e.g., defect-proneness) while having the lowest correlation among each other. To better understand whether feature selection techniques mitigate correlated metrics, we set out to perform a correlation analysis on the metrics that are selected by feature selection techniques. In this preliminary analysis (PA4), we focus on the two commonly-used techniques in the domain of defect prediction, i.e., Information Gain and correlation-based feature selection techniques. The results of this preliminary analysis show that the metrics that are selected by the two studied feature selection techniques are correlated (with a Spearman correlation coefficient up to 0.98), suggesting that the commonly-used feature selection techniques do not mitigate correlated metrics.

Correlation analysis is used to measure the correlation among metrics given a threshold. Prior work applies correlation analysis techniques to identify and mitigate correlated metrics [15, 68, 82, 83]. Based on a literature survey of Hall *et al.* [26] and Shihab [67], we select the commonly-used correlation analysis techniques: Variable Clustering analysis (VarClus), and Variance Inflation Factor (VIF).

Since there are many analytical learners that can be used to investigate the impact of correlated metrics on defect models, the aforementioned surveys guide our selection of the two commonly-used analytical learners: logistic regression [5, 6, 15, 43, 53, 57, 58, 65, 87] and random forest [23, 24, 38, 55, 64]. These techniques are two of the most commonly-used analytical learners for defect models and they have built-in techniques for model interpretation

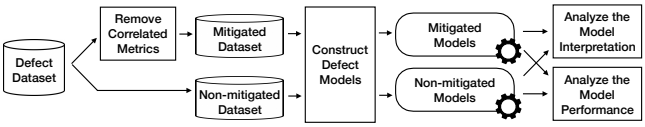


Figure 2: An overview diagram of the design of our case study.

Table 1: A statistical summary of the studied datasets.

Project	Dataset	Modules	Metrics	Correlated Metrics	EPV	AUC_{LR}	AUC_{RF}
Apache	Lucene 2.4	340	20	9	10	0.74	0.77
	POI 2.5	385	20	11	12	0.80	0.90
	POI 3.0	442	20	10	14	0.79	0.88
	Xalan 2.6	885	20	8	21	0.79	0.85
	Xerces 1.4	588	20	11	22	0.91	0.95
Eclipse	Debug 3.4	1,065	17	9	15	0.72	0.81
	JDT	997	15	10	14	0.81	0.82
	Mylyn	1,862	15	10	16	0.78	0.74
	PDE	1,497	15	9	14	0.72	0.72
	Platform 2.0	6,729	32	24	30	0.82	0.84
	Platform 3.0	10,593	32	24	49	0.79	0.81
	SWT 3.4	1,485	17	7	38	0.87	0.97
Proprietary	Prop 1	18,471	20	10	137	0.75	0.79
	Prop 4	8,718	20	11	42	0.74	0.72

(i.e., ANOVA for logistic regression and Breiman's Variable Importance for random forest). Finally, we select 9 model interpretation techniques, ANOVA Type-I, ANOVA Type-II with 4 test statistics (i.e., Wald, Likelihood Ratio, F, and Chi-square), scaled and non-scaled Gini Importance, and scaled and non-scaled Permutation Importance. We provide the detailed explanation of the studied correlation analysis techniques, analytical learners, and interpretation techniques in Table 2 and Appendix 1 of the supplementary materials.

3 Case Study Design

In this study, we use 14 datasets of systems that span across proprietary and open-source systems. We discuss the selection criteria of the studied datasets in Appendix 3 of the supplementary materials. Table 1 shows a statistical summary of the studied datasets, while Figure 2 provides an overview of the design of our case study. Below, we discuss the design of the case study that we perform in order to address our four research questions.

3.1 Remove Correlated Metrics

To investigate the impact of correlated metrics on the performance and interpretation of defect models and address our four research questions, we start by removing highly-correlated metrics in order to produce mitigated datasets, i.e., datasets where correlated metrics are removed. To do so, we apply variable clustering analysis (VarClus) and variable influence factor analysis (VIF) (see Appendix 1.1). We use the interpretation of Spearman correlation coefficients ($|\rho|$) as provided by Kraemer *et al.* [45] to identify correlated metrics, i.e., a Spearman correlation coefficient of above 0.7 is considered a strong correlation. We use a VIF threshold of 5 to identify inter-correlated metrics, as it is suggested by Fox [18] and is commonly used in prior work [4, 50, 68, 69]. We use the implementation of the variable clustering analysis as provided by the `varclus` function of the `Hmisc` R package [28]. We use the implementation of the VIF analysis as provided by the `vif` function of the `rms` R package [30]. Finally, we report the results of correlation analysis and a set of mitigated metrics for each of the studied defect datasets in the online appendix [34].

3.2 Construct Defect Models

To examine the impact of correlated metrics on the performance and interpretation of defect models, we construct our models using the *non-mitigated datasets* (i.e., datasets where correlated metrics are not removed) and *mitigated datasets* (i.e., datasets where correlated metrics are removed). To construct defect models, we perform the following steps:

(CM1) Generate bootstrap samples. To ensure that our conclusions are statistically sound and robust, we use the out-of-sample bootstrap validation technique, which leverages aspects of statistical inference [17, 19, 29, 72, 78]. We first generate bootstrap sample of sizes N with replacement from the mitigated and non-mitigated datasets. The generated sample is also of size N. We construct models using the bootstrap samples, while we measure the performance of the models using the samples that do not appear in the bootstrap samples. On average, 36.8% of the original dataset will not appear in the bootstrap samples, since the samples are drawn with replacement [17]. We repeat the out-of-sample bootstrap process for 100 times and report their average performance.

(CM2) Construct defect models. For each bootstrap sample, we construct logistic regression and random forest models. We use the implementation of logistic regression as provided by the `glm` function of the `stats` R package [80] and the `lrm` function of the `rms` R package [30] with the default parameter setting. We use the implementation of random forest as provided by the `randomForest` function of the `randomForest` R package [9] with the default `ntree` value of 100, since recent studies [76, 79] show that parameters of random forest are insensitive to the performance of defect models. To ensure that the training and testing corpora share similar characteristics and representative to the original dataset, we do not re-balance nor do we re-sample the training data to avoid any impact on the interpretation of defect models [75].

3.3 Analyze the Model Interpretation

To address RQ1, RQ2, and RQ3, we analyze the importance ranking of metrics of the models that are constructed using non-mitigated datasets and mitigated datasets. The analysis of model interpretation is made up of 2 steps.

(MI1) Compute the importance score of metrics. We investigate the impact of correlated metrics on the interpretation of defect models using different model interpretation techniques. Thus, we apply the 9 studied model interpretation techniques, i.e., Type-I, Type-II (Wald, LR, F, Chisq), scaled and non-scaled Gini Importance, and scaled and non-scaled Permutation Importance. We provide the technical description of the studied interpretation techniques in Appendix 1.3 of the supplementary materials.

(MI2) Identify the importance ranking of metrics. To statistically identify the importance ranking of metrics, we apply the improved Scott-Knott Effect Size Difference (ESD) test (v2.0) [73]. The Scott-Knott ESD test is a mean comparison approach that leverages a hierarchical clustering to partition a set of treatment means (i.e., means of

Table 2: A summary of the studied correlation analysis techniques, the two studied analytical learners, and the 9 studied interpretation techniques.

Correlation Analysis	Analytical Learner	Interpretation Technique	Test Statistic	R function
Variable Clustering [56, 81-83]	Logistic Regression (`glm` and `lrm`) [5, 6, 57, 58, 87]	Type-I	Deviance	`stats::anova(glm.model)`
		Type-II	Wald	`car::Anova(glm.model, type=2, test.statistic='Wald')`
			Likelihood Ratio (LR)	`car::Anova(glm.model, type=2, test.statistic='LR')`
Variance Inflation Factor [4, 15, 50, 68, 69]			F	`car::Anova(glm.model, type=2, test.statistic='F')`
			Chi-square	`rms::anova(lrm.model, test='Chisq')`
Redundancy Analysis [2, 37, 56, 70, 77]	Random Forest [23, 24, 38, 55, 64]	Scaled Gini	MeanDecreaseGini	`randomForest::importance(model, type = 2, scale = TRUE)`
		Non-scaled Gini	MeanDecreaseGini	`randomForest::importance(model, type = 2, scale = FALSE)`
		Scaled Permutation	MeanDecreaseAccuracy	`randomForest::importance(model, type = 1, scale = TRUE)`
		Non-scaled Permutation	MeanDecreaseAccuracy	`randomForest::importance(model, type = 1, scale = FALSE)`

importance scores) into statistically distinct groups *with statistically non-negligible difference*. The Scott-Knott ESD test ranks each metric at only a single rank, however several metrics may appear within one rank. Finally, we identify the importance ranking of metrics for the non-mitigated and mitigated models. Thus, each metric has a rank for each model interpretation technique and for each of the mitigated and non-mitigated models. We use the implementation of Scott-Knott ESD test as provided by the `sk_esd` function of the `ScottKnottESD` R package [73].

3.4 Analyze the Model Performance

To address RQ4, we analyze the performance of the models that are constructed using non-mitigated datasets and mitigated datasets.

First, we use the Area Under the receiver operator characteristic Curve (AUC) to measure the discriminatory power of our models, as suggested by recent research [21, 46, 62]. The AUC is a threshold-independent performance measure that evaluates the ability of models in discriminating between defective and clean modules. The values of AUC range between 0 (worst performance), 0.5 (no better than random guessing), and 1 (best performance) [27].

Second, we use the F-measure, i.e, a threshold-dependent measure. F-measure is a harmonic mean (i.e., $\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision}+\text{recall}}$) of precision ($\frac{\text{TP}}{\text{TP}+\text{FP}}$) and recall ($\frac{\text{TP}}{\text{TP}+\text{FN}}$). Similar to prior studies [1, 86], we use the default probability value of 0.5 as a threshold value for the confusion matrix, i.e., if a module has a predicted probability above 0.5, it is considered defective; otherwise, the module is considered clean.

Third, we use the Matthews Correlation Coefficient (MCC) measure, i.e, a threshold-dependent measure, as suggested by prior studies [48, 66]. MCC is a balanced measure based on true and false positives and negatives that is computed using the following equation: $\frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP}+\text{FP})(\text{TP}+\text{FN})(\text{TN}+\text{FP})(\text{TN}+\text{FN})}}$.

4 Case Study Results

In this section, we present the results of our case study with respect to our four research questions.

(RQ1) How do correlated metrics impact the interpretation of defect models?

Motivation. Prior work raises concerns that metrics are often correlated and should be mitigated [22, 32, 33, 35, 74, 77, 85]. For example, Herraiz *et al.* [33], and Gil *et al.* [22] point out that code complexity is often correlated with lines of code. Unfortunately, a literature survey of Shihab [67] shows that as much as 63% of prior defect studies do not mitigate correlated metrics prior to constructing defect models. Yet, little is known about the impact of correlated metrics on the interpretation of defect models.

Approach. To address RQ1, we analyze the impact of correlated metrics on the interpretation of defect models along with three dimensions, i.e., (1) the *consistency* of the top-ranked metric, (2) the *level of discrepancy* of the top-ranked metric, and (3) the *direction* of the ranking of metrics for all non-correlated metrics between mitigated and non-mitigated models.

To do so, we start from mitigated datasets (see Section 3.1). We first identify the top-ranked metric for each of the 9 studied interpretation techniques. We use VarClus to select only one of the metrics that is correlated with the top-ranked metric in order to generate non-mitigated datasets. We then append the correlated metric to the first position of the specification of the mitigated models. Thus, the specification for the mitigated models is $y \sim m_{\text{top\_ranked}} + ...$, while the specification for the non-mitigated models is $y \sim m_{\text{correlated}} + m_{\text{top\_ranked}} + ...$, where $m_{\text{correlated}}$ is the metric that is correlated with the top-ranked metric ($m_{\text{top\_ranked}}$). For each of the mitigated and non-mitigated datasets, we construct defect models (see Section 3.2) and apply the 9 studied model interpretation techniques (see Section 3.3).

To analyze the *consistency* and the *level of discrepancy* of the top-ranked metric, we compute the difference in the ranks of the top-ranked metric between mitigated and non-mitigated models. For example, if a metric $m_{\text{top\_ranked}}$ appears in the 1^{st} rank in both of mitigated and non-mitigated models, then the metric would have a rank difference of 0. However, if $m_{\text{top\_ranked}}$ appears in the 3^{rd} rank of a non-mitigated model and appears in the 1^{st} rank of a mitigated model, then the rank difference of $m_{\text{top\_ranked}}$ would be 2. The consistency of the top-ranked metric measures the percentage of the studied datasets that the top-ranked metric appears at the 1^{st} rank in both mitigated and non-mitigated models. On the other hand, the level of discrepancy of the top-ranked metric measures the highest rank difference of the top-ranked metric between mitigated and non-mitigated models.

To analyze the *direction* of the ranking of metrics for all non-correlated metrics between mitigated and non-mitigated models, we start with the ranking of important

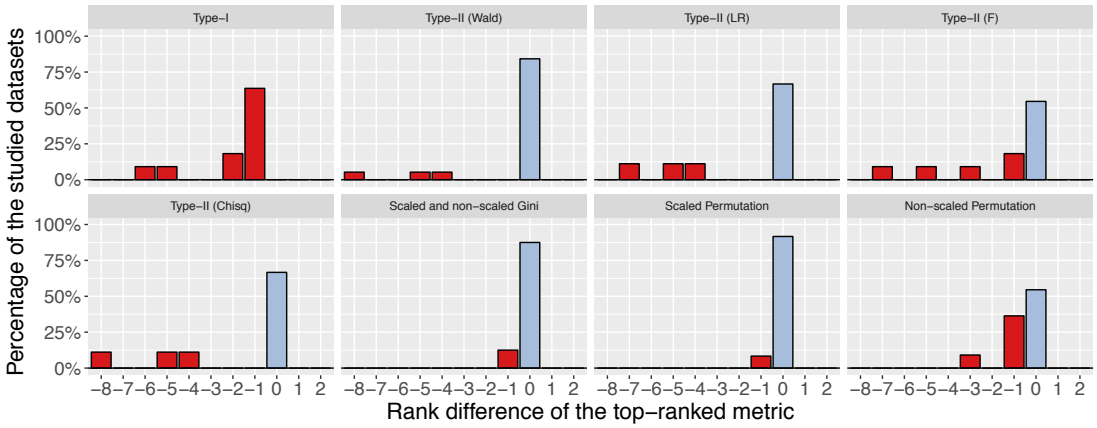


Figure 3: The percentage of the studied datasets for each difference in the ranks between the top-ranked metric of the models that are constructed using the mitigated and non-mitigated datasets. The light blue bars represent the consistent rank of the metric between mitigated and non-mitigated models, while the red bars represent the inconsistent rank of the metric between mitigated and non-mitigated models.

metrics that appear in both mitigated and non-mitigated models. We then apply a Spearman rank correlation test (ρ) to compute the correlation between ranks of all non-correlated metrics between mitigated and non-mitigated models. The Spearman correlation coefficient (ρ) of 1 indicates that the ranking of non-correlated metrics between mitigated and non-mitigated models is in the same direction. On the other hand, the Spearman correlation coefficient (ρ) of -1 indicates that the ranking of non-correlated metrics between mitigated and non-mitigated models is in the reverse direction. Since the produced ranking of each model and defect dataset may be different, the use of a weighted Spearman rank correlation test may lead these rankings to be weighted differently. Thus, we select a traditional Spearman rank correlation test for our study. We report the distributions of the Spearman correlation coefficients (ρ) of the ranking of metrics between mitigated and non-mitigated models for all studied interpretation techniques in Figure S7 in the supplementary materials.

Results. **ANOVA Type-I produces the lowest consistency of the top-ranked metric between mitigated and non-mitigated models.** We expect that the top-ranked metric in the mitigated model will remain as the top-ranked metric in the non-mitigated model. Unfortunately, Figure 3 shows that this expectation does not hold true in any of the studied datasets for ANOVA Type-I. Figure 3 shows that, for ANOVA Type-I, none of the top-ranked metric that appears in the 1^{st} rank of mitigated models also appears in the 1^{st} rank of non-mitigated models. On the other hand, we find that the top-ranked metric of mitigated models appears at the top-ranked metric in non-mitigated models for 84%, 67%, 55%, and 67% of the studied datasets for ANOVA Type-II (Wald), Type-II (LR), Type-II (F), Type-II (Chisq), respectively. We suspect that the impact of correlated metrics on the interpretation of Type-I has to do with the sequential nature of the calculation of the Sum of Squares, i.e., Type-I attributes as much variance as it can to the first metric before attributing residual variance to the second metric in the model specification.

ANOVA Type-I and Type-II produce the highest level of discrepancy of the top-ranked metrics between mitigated and non-mitigated models. Figure 3 shows that the rank difference for ANOVA Type-I and Type-II can be up to -6 and -8, respectively. In other words, we find that the top-ranked metric in mitigated models appear at the 7^{th} rank and the 9^{th} rank in non-mitigated models for ANOVA Type-I and Type-II, respectively. We suspect that the highest level of discrepancy (i.e., the largest rank difference) for ANOVA Type-I and Type-II has to do with the sharply drop of the importance scores when correlated metrics are included in defect models (see PA1).

For ANOVA Type-I and Type-II, correlated metrics have the largest impact on the direction of the ranking of metrics of defect models. For ANOVA Type-I, we find that the Spearman correlation coefficients range from -0.1 to 0.84 (see Figure S7). For ANOVA Type-II, we find that the Spearman correlation coefficients range from 0.1 to 1. A low value of the Spearman correlation coefficients in ANOVA techniques indicates that the direction of the ranking of metrics for all non-correlated metrics is varied in each rank, suggesting that the ranking of non-correlated metrics is inconsistent across mitigated and non-mitigated models.

Gini and Permutation Importance approaches produce the higest consistency and the lowest level of discrepancy of the top-ranked metric, and have the least impact on the direction of the ranking of metrics between mitigated and non-mitigated models. Figure 3 shows that the top-ranked metric of mitigated models appears at the top-ranked metric in non-mitigated models for 88%, 92%, and 55% of the studied datasets for Gini Importance, and scaled and non-scaled Permutation Importance, respectively. Figure 3 also shows that the rank difference for Gini and Permutation Importance is as low as -1 and -3, respectively. Furthermore, we find that the Spearman correlation coefficients range from 0.9 to 1 for Gini and Permutation Importance (see Figure S7). These findings suggest that the lower impact that correlated metrics have on Gini and Permutation Importance than ANOVA techniques have to do with the random process for constructing multiple trees and the calculation of importance scores for a random forest model. For example,

the random process of random forest may generate trees that are constructed without correlated metrics. In addition, the averaging of the importance scores from multiple trees may decrease the negative impact of correlated metrics on trees that are constructed with correlated metrics for random forest models.

ANOVA Type-I and Type-II often produce the lowest consistency and the highest level of discrepancy of the top-ranked metric, and have the highest impact on the direction of the ranking of metrics between mitigated and non-mitigated models when compared to Gini and Permutation Importance. This finding highlights the risks of not mitigating correlated metrics in the ANOVA analyses of prior studies.

(RQ2) After removing all correlated metrics, how consistent is the interpretation of defect models among different model specifications?

Motivation. Our motivating analysis (see Appendix 2) and the results of RQ1 confirm that the ranking of the top-ranked metric substantially changes when the ordering of correlated metrics in a model specification is rearranged, suggesting that correlated metrics must be removed. However, after removing correlated metrics, little is known if the interpretation of defect models would become consistent when rearranging the ordering of metrics.

Approach. To address RQ2, we analyze the ranking of the top-ranked metric of the models that are constructed using different ordering of metrics from mitigated datasets. To do so, we start from mitigated datasets that are produced by Section 3.1. For each of the datasets, we construct defect models (see Section 3.2) and apply the 9 studied model interpretation techniques (see Section 3.3) in order to identify the top-ranked metric according to each technique. Then, we regenerate the models where the ordering of metrics is rearranged—the top-ranked metric is at each position from the first to the last for each dataset. Finally, we compute the percentage of datasets where the ranks of the top-ranked metric are inconsistent among the rearranged datasets.

Results. **After removing correlated metrics, the top-ranked metrics according to Type-II, Gini Importance, and Permutation Importance are consistent. However, the top-ranked metric according to Type-I is still inconsistent regardless of the ordering of metrics.** We find that Type-II, Gini Importance, and Permutation Importance produce a stable ranking of the top-ranked metric for all of the studied datasets regardless of the ordering of metrics.

On the other hand, ANOVA Type-I is the only technique that produces an inconsistent ranking of the top-ranked metric. We find that for 71% of the studied datasets, ANOVA Type-I produces an inconsistent ranking of the top-ranked metric when the ordering of metrics is rearranged. We expect that the consistency of the ranking of the top-ranked metrics can be improved by increasing the strictness of the correlation threshold of the variable clustering analysis (VarClus). Thus, we repeat the analysis using stricter thresholds of the variable clustering analysis (VarClus). We use Spearman correlation coefficient ($|\rho|$) threshold values of 0.5 and 0.6. Unfortunately, even if we increase the strictness of the correlation threshold value, Type-I produces the inconsistent ranking of the top-ranked metric for 43% and 50% of the studied datasets, for the threshold of 0.5 and 0.6, respectively.

The inconsistent ranking of the top-ranked metric according to Type-I has to do with the sequential nature of the calculation of the Sum of Squares (see Appendix 1.3). In other words, Type-I attributes the importance scores as much as it can to the first metric before attributing the scores to the second metric in the model specification. Thus, Type-I is sensitive to the ordering of metrics.

After removing all correlated metrics, the top-ranked metric according to ANOVA Type-II, Gini Importance, and Permutation Importance are consistent. However, the top-ranked metric according to ANOVA Type-I is inconsistent, since the ranking of metrics is impacted by its order in the model specification when analyzed using ANOVA Type-I (which is the default analysis for the `glm` model in R and is commonly-used in prior studies). This finding suggests that ANOVA Type-I must be avoided even if all correlated metrics are removed.

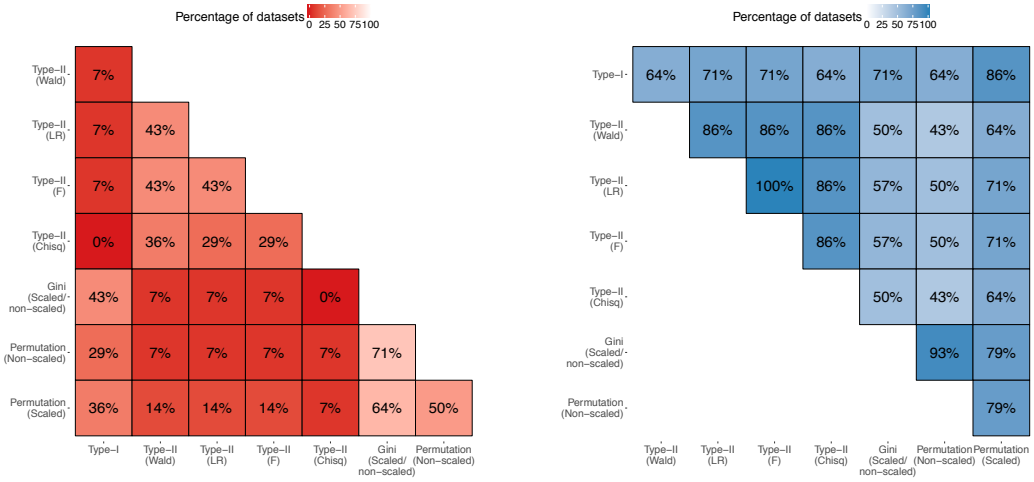
(RQ3) After removing all correlated metrics, how consistent is the interpretation of defect models among the studied interpretation techniques?

Motivation. The findings of prior work often rely heavily on one model interpretation technique [23, 37, 55, 56, 70, 77]. Therefore, the findings of prior work may pose a threat to construct validity, i.e., the findings may not hold true if one uses another interpretation technique. Thus, we set out to investigate the consistency of the top-ranked and top-3 ranked metrics after removing correlated metrics.

Approach. To address RQ3, we start from mitigated datasets that are produced by Section 3.1 and non-mitigated datasets (i.e., the original datasets). We compare the two rankings that are produced from mitigated and non-mitigated models using the 9 interpretation techniques for each of the studied datasets. Then, we compute the percentage of datasets where the top-ranked metric is consistent among the studied model interpretation techniques. Moreover, we also compute the percentage of datasets where at least one of the top-3 ranked metrics is consistent among the studied model interpretation techniques. Finally, we present the results using a heatmap (as shown in Figure 4) where each cell indicates the percentage of datasets which the top-ranked metric is consistent among the two studied model interpretation techniques.

Results. **Before removing all correlated metrics, we find that the studied model interpretation techniques do not tend to produce the same top-ranked metric.** We observe that the consistency of the ranking of metrics across learning techniques (i.e., logistic regression and random forest) is as low as 0%-43% for the top-ranked metric (Figure 4a) and 21%-57% for the top-3 ranked metrics (see Figure S8a), respectively. Furthermore, according to the lower-left side of the matrix of the Figure 4a, we find that, before removing correlated metrics, the top-ranked metric of Type-II (Chisq) is inconsistent with the top-ranked metrics of Type-I and Gini Importance for all of the studied datasets.

After removing all correlated metrics, we find that the consistency of the ranking of metrics among the studied interpretation techniques is improved by 15%-64% for



(a) The top-ranked metric for non-mitigated models. (b) The top-ranked metric for mitigated models.

Figure 4: The percentage of datasets where the top-ranked metric is consistent between the two studied model interpretation techniques. While the lower-left side of the matrix (i.e., red shades) shows the percentage before removing correlated metrics, the upper-right side of the matrix (i.e., blue shades) shows the percentage after removing correlated metrics. For the consistency of the top-3 ranked metrics, see Figure S8 in the supplementary materials.

the top-ranked metric and 21%-71% for the top-3 ranked metrics, respectively. In particular, we observe that the consistency of the ranking of metrics across learning techniques is improved by 28%-50% for the top-1 ranked metrics and 43%-71% for the top-3 ranked metrics, respectively. Most importantly, we find that scaled Permutation Importance achieves the highest consistency of the ranking of metrics across learning techniques. This finding highlights the benefits of removing correlated metrics on the interpretation of defect models—the conclusions of studies that rely on one interpretation technique may not pose a threat after mitigating correlated metrics.

After removing all correlated metrics, we find that the consistency of the ranking of metrics among the studied interpretation techniques is improved by 15%- 64% for the top-ranked metric and 21%-71% for the top-3 ranked metrics, respectively, highlighting the benefits of removing all correlated metrics on the interpretation of defect models, i.e., the conclusions of studies that rely on one interpretation technique may not pose a threat after mitigating correlated metrics.

(RQ4) Does removing all correlated metrics impact the performance and stability of defect models?

Motivation. The results of RQ1 show that correlated metrics have a negative impact on the interpretation of defect prediction models, while the results of RQ2 and RQ3 show the benefits of removing correlated metrics on the interpretation of defect models. Thus, removing correlated metrics is highly recommended. However, removing correlated metrics may pose a risk to the performance and stability of defect models. Yet, little is known if removing such correlated metrics impacts the performance and stability of defect models.

Approach. To address RQ4, we first start from the AUC, F-measure and MCC performance estimates and their performance stability of the non-mitigated and mitigated models. The performance stability is measured by a standard deviation of the performance estimates as produced by 100 iterations of the out-of-sample bootstrap for each model. We then quantify the impact of removing all correlated metrics by measuring the performance difference (i.e., the arithmetic difference between the performance of the non-mitigated and mitigated models) and the stability ratio (i.e., the ratio of the $S.D.$ of performance estimates of non-mitigated to mitigated models, $\frac{S.D.\text{ non-mitigated models}}{S.D.\text{ mitigated models}}$). Furthermore, in order to measure the effect size of the impact, we measure Cliff's $|\delta|$ effect size for the performance difference and the stability ratio across the non-mitigated and mitigated models.

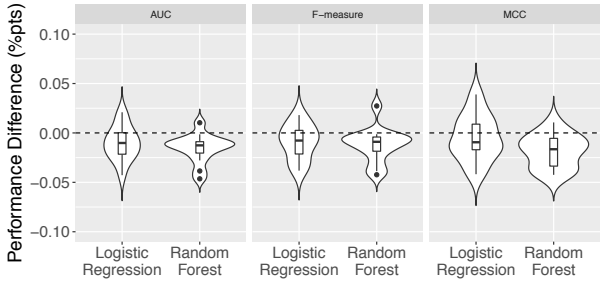


Figure 5: The distributions of the performance difference (% pts) between non-mitigated and mitigated models for each of the studied datasets.

Results. **Removing all correlated metrics impacts the AUC, F-measure, and MCC performance of defect models by less than 5 percentage points.** Figure 5 shows that the distributions of the performance difference between the models that are constructed using non-mitigated and mitigated datasets are centered at zero. In addition, our Cliff's $|\delta|$ effect size test shows that the differences between the models that are constructed using mitigated and non-

mitigated datasets are negligible to small for the AUC, F-measure, and MCC measures. However, the performance difference of 5 percentage points may be very important for safety-critical software domains. Thus, researchers and practitioners should remove correlated metrics with care.

Removing all correlated metrics yields a negligible difference (Cliff's $|\delta|$) for the stability of the performance of defect models. Figure 6 shows that the distributions of the stability ratio of the models that are constructed using non-mitigated and mitigated datasets are centered at one (i.e., there is little difference in model stability after removing all correlated metrics). Moreover, our Cliff's $|\delta|$ effect size test shows that the difference of the stability ratio between the models that are constructed using mitigated and non-mitigated datasets is negligible.

Removing all correlated metrics impacts the AUC, F-measure, and MCC performance of defect models by less than 5 percentage points, suggesting that researchers and practitioners should remove correlated metrics with care especially for safety-critical software domains.

5 Practical Guidelines

In this section, we offer practical guidelines for future studies. When the goal is to derive sound interpretation from defect models:

(1) Ones must mitigate correlated metrics prior to constructing a defect model, especially for ANOVA analyses, since RQ1 shows that (1) ANOVA Type-I and Type-II often produce the lowest consistency and the highest level of discrepancy of the top-ranked metric, and have the highest impact on the direction of the ranking of metrics between mitigated and non-mitigated models. On the other hand, the results of RQ2 and RQ3 show that removing all correlated metrics (2) improves the consistency of the top-ranked metric regardless of the ordering of metrics; and (3) improves the consistency of the ranking of metrics among to the studied interpretation techniques, suggesting that correlated metrics must be mitigated. However, the results of RQ4 show that the removal of such correlated metrics impacts the model performance by less than 5 percentage points, suggesting that researchers and practitioners should remove correlated metrics with care especially for safety-critical software domains.

(2) Ones must avoid using ANOVA Type-I even if all correlated metrics are removed, since RQ2 shows that Type-I produces an inconsistent ranking of the top-ranked metric when the orders of metrics are rearranged, indicating that Type-I is sensitive to the ordering of metrics even when removing all correlated metrics. Instead, researchers should opt to **use ANOVA Type-II and Type-III for additive and interaction logistic regression models, respectively**. Furthermore, the scaled Permutation Importance approach is recommended for random forest since RQ3 shows that such approach achieves the highest consistency across learning techniques.

Finally, we would like to emphasize that mitigating correlated metrics is not necessary for all studies, all scenarios, all datasets, and all analytical models in software engineering. Instead, the key message of our study is to shed light that correlated metrics must be mitigated when the goal is to derive sound interpretation from defect models that are trained with correlated metrics (especially for ANOVA Type-I). On the other hand, if the goal of the study is to produce highly-accurate prediction models, one might prioritize their resources on improving the model performance rather than mitigating correlated metrics. Thus, feature selection and dimensionality reduction techniques can be considered to mitigate irrelevant and correlated metrics, and improve model performance.

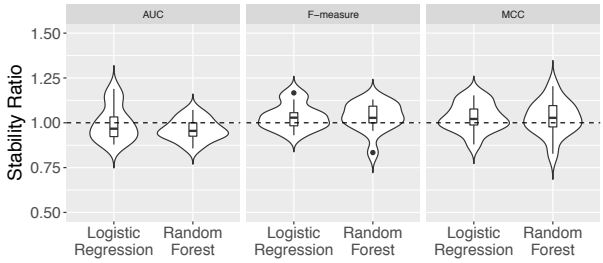


Figure 6: The distributions of the stability ratio of non-mitigated to mitigated models for each of the studied datasets.

6 Threats to Validity

Construct Validity. In this work, we only construct regression models in an additive fashion ($y \sim m_1 + ... + m_n$), since metric interactions (i.e., the relationship between each of the two interacting metrics depends on the value of the other metrics) (1) are rarely explored in software engineering (except in [66]); (2) must be statistically insignificant (e.g., absence) for ANOVA Type-II test [13, 18]; and (3) are not compatible with random forest [8] which is one of the most commonly-used analytical learners in software engineering. On the other hand, the importance score of the metric produced by ANOVA Type-III is evaluated after all of the other metrics and all metric interactions of the metric under examination have been accounted for. Thus, if metric interactions are significantly present, one should use ANOVA Type-III and avoid using ANOVA Type-II. Due to the same way in which the importance scores of metrics according to ANOVA Type-II and Type-III are calculated in a hierarchical nature (see Appendix 1.3) for an additive model, we would like to note that the importance scores of metrics according to ANOVA Type-II and Type-III are the same for such additive models.

Plenty of prior work show that the parameters of classification techniques have an impact on the performance of defect models [20, 44, 51, 52, 76, 79]. While we use a default `ntree` value of 100 for random forest models, recent studies [76, 79] show that the parameters of random forest are insensitive to the performance of defect models. Thus, the parameters of random forest models do not pose a threat to validity of our study.

Recent work point out that the selection [39, 76] and the quality [84] of datasets dataset selection might impact conclusions of a study. Thus, our conclusions may alter

when changing a set of the studieds datasets. Moreover, Tantithamthavorn *et al.* [78] point out that randomness may introduce bias to the conclusions of a study. To mitigate this threat and ensure that our results are reproducible, we set a random seed in every step in our experiment design.

Internal Validity. Recent research uses ridge regression to construct defect models on the dataset that contains correlated metrics [63]. However, our additional analyses (Appendix 4 of the supplementary materials) show that ridge regression improves the performance of defect model when comparing to logistic regression, yet produces a misleading importance ranking of metrics. We observe that metrics that are highly correlated appear at different ranks. This observation highlights the importance of mitigating correlated metrics when interpreting defect models.

We studied a limited number of model interpretation techniques. Thus, our results may not generalize to other model interpretation techniques. Nonetheless, other model interpretation techniques can be explored in future work. We provide a detailed methodology for others who would like to re-examine our findings using unexplored model interpretation techniques.

External Validity. The analyzed datasets are part of several corpora (e.g., NASA and PROMISE) of systems that span both proprietary and open source domains. However, we studied a limited number of defect datasets. Thus, the results may not generalize to other datasets and domains. Nonetheless, additional replication studies are needed.

In our study, we exclude (1) datasets that are not representative of common practice or (2) datasets that would not realistically benefit from our analysis (e.g., datasets that most of the software modules are defective) with the selection criteria of the studied datasets (in Appendix 3). Nevertheless, our proposed approaches are applicable to any dataset. Practitioners are encouraged to explore our approaches on their own datasets with their own peculiarities.

The conclusions of our case study rely on one defect prediction scenario (i.e., within-project defect models). However, there are a variety of defect prediction scenarios in the literature (e.g., cross-project defect prediction [12, 86], just-in-time defect prediction [38], heterogenous defect prediction [60]). Therefore, the practical guidelines may differ for other scenarios. Thus, future research should revisit our study in other scenarios of defect models.

7 Conclusion

In this paper, we set out to investigate (1) the impact of correlated metrics on the interpretation of defect models. After removing correlated metrics, we investigate (2) the consistency of the interpretation of defect models; and (3) its impact on the performance and stability of defect models. Through a case study of 14 publicly-available defect datasets of systems that span both proprietary and open source domains, we conclude that (1) correlated metrics have the largest impact on the consistency, the level of discrepancy, and the direction of the ranking of metrics, especially for ANOVA techniques. On the other hand, we find that removing all correlated metrics (2) improves the consistency of the produced rankings regardless of the ordering of metrics (except for ANOVA Type-I); (3) improves the consistency of ranking of metrics among the studied interpretation techniques; (4) impacts the model performance by less than 5 percentage points.

Based on our findings, we offer practical guidelines for future studies. When the goal is to derive sound interpretation from defect models:

1) Ones must mitigate correlated metrics prior to constructing a defect model, especially for ANOVA analyses.
2) Ones must avoid using ANOVA Type-I even if all correlated metrics are removed.

Due to the variety of the built-in interpretation techniques and their settings, our paper highlights the essential need for future research to report the exact specification of their models and settings of the used interpretation techniques.

References

[1] E. Arisholm, L. C. Briand, and E. B. Johannessen, "A Systematic and Comprehensive Investigation of Methods to Build and Evaluate Fault Prediction Models," *Journal of Systems and Software*, vol. 83, no. 1, pp. 2–17, 2010.

[2] J. G. Barnett, C. K. Gathuru, L. S. Soldano, and S. McIntosh, "The Relationship between Commit Message Detail and Defect Proneness in Java Projects on GitHub," in *Proceedings of the International Conference on Mining Software Repositories (MSR)*, 2016, pp. 496–499.

[3] V. R. Basili, L. C. Briand, and W. L. Melo, "A Validation of Object-oriented Design Metrics as Quality Indicators," *Transactions on Software Engineering (TSE)*, vol. 22, no. 10, pp. 751–761, 1996.

[4] N. Bettenburg and A. E. Hassan, "Studying the Impact of Social Structures on Software Quality," in *Proceedings of the International Conference on Program Comprehension (ICPC)*, 2010, pp. 124–133.

[5] C. Bird, B. Murphy, and H. Gall, "Don't Touch My Code ! Examining the Effects of Ownership on Software Quality," in *Proceedings of the Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2011, pp. 4–14.

[6] C. Bird, N. Nagappan, P. Devanbu, H. Gall, and B. Murphy, "Does Distributed Development Affect Software Quality?: An Empirical Case Study of Windows Vista," *Communications of the ACM*, vol. 52, no. 8, pp. 85–93, 2009.

[7] D. Bowes, T. Hall, M. Harman, Y. Jia, F. Sarro, and F. Wu, "Mutation-Aware Fault Prediction," in *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)*, 2016, pp. 330–341.

[8] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] L. Breiman, A. Cutler, A. Liaw, and M. Wiener, "randomForest : Breiman and Cutler's Random Forests for Classification and Regression. R package version 4.6-12." *Software available at URL: https://cran.r-project.org/web/packages/randomForest*, 2006.

[10] L. C. Briand, W. L. Melo, and J. Wust, "Assessing the Applicability of Fault-proneness Models across Object-oriented Software Projects," *Transactions on Software Engineering (TSE)*, vol. 28, no. 7, pp. 706–720, 2002.

[11] L. C. Briand, J. Wüst, J. W. Daly, and D. V. Porter, "Exploring the Relationships between Design Measures and Software Quality in Object-oriented Systems," *Journal of Systems and Software*, vol. 51, no. 3, pp. 245–273, 2000.

[12] G. Canfora, A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella, "Multi-objective Cross-project Defect Prediction," in *Proceedings of the International Conference on Software Testing, Verification and Validation (ICST)*, 2013, pp. 252–261.

[13] J. M. Chambers, "Statistical Models in S. Wadsworth," *Pacific Grove, California*, 1992.

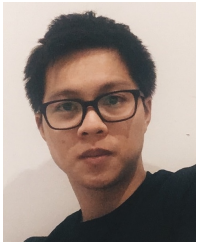
[14] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," *Transactions on Software Engineering (TSE)*, vol. 20, no. 6, pp. 476–493, 1994.

[15] P. Devanbu, T. Zimmermann, and C. Bird, "Belief & Evidence in Empirical Software Engineering," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2016, pp. 108–119.

[16] M. Di Penta, L. Cerulo, Y.-G. Guéhéneuc, and G. Antoniol, "An Empirical Study of the Relationships between Design Pattern

Roles and Class Change Proneness," in *Proceedings of the International Conference on Software Maintenance (ICSM)*, 2008, pp. 217–226.
[17] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. Boston, MA: Springer US, 1993.
[18] J. Fox, *Applied regression analysis and generalized linear models*. Sage Publications, 2015.
[19] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning*. Springer series in statistics, 2001, vol. 1.
[20] W. Fu, T. Menzies, and X. Shen, "Tuning for Software Analytics: Is it really necessary?" *Information and Software Technology*, vol. 76, pp. 135–146, 2016.
[21] B. Ghotra, S. McIntosh, and A. E. Hassan, "Revisiting the Impact of Classification Techniques on the Performance of Defect Prediction Models," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2015, pp. 789–800.
[22] Y. Gil and G. Lalouche, "On the Correlation between Size and Metric Validity," *Empirical Software Engineering (EMSE)*, vol. 22, no. 5, pp. 2585–2611, 2017.
[23] G. Gousios, M. Pinzger, and A. v. Deursen, "An Exploratory Study of the Pull-based Software Development Model," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2014, pp. 345–355.
[24] L. Guo, Y. Ma, B. Cukic, and H. Singh, "Robust Prediction of Fault-proneness by Random Forests," in *Proceedings of the International Symposium on Software Reliability Engineering (ISSRE)*, 2004, pp. 417–428.
[25] M. A. Hall and L. A. Smith, "Feature Subset Selection: A Correlation Based Filter Approach," 1997.
[26] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A Systematic Literature Review on Fault Prediction Performance in Software Engineering," *Transactions on Software Engineering (TSE)*, vol. 38, no. 6, pp. 1276–1304, 2012.
[27] J. a. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve." *Radiology*, vol. 143, no. 4, pp. 29–36, 1982.
[28] F. E. Harrell Jr, "Hmisc: Harrell miscellaneous. R package version 3.12-2," *Software available at URL: http://cran.r-project.org/web/packages/Hmisc*, 2013.
[29] ——, *Regression Modeling Strategies : With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*. Springer, 2015.
[30] ——, "rms: Regression Modeling Strategies. R package version 5.1-1," 2017.
[31] A. E. Hassan, "Predicting Faults using the Complexity of Code Changes," in *Prooceedings of the International Conference on Software Engineering (ICSE)*, 2009, pp. 78–88.
[32] H. Hemmati, S. Nadi, O. Baysal, O. Kononenko, W. Wang, R. Holmes, and M. W. Godfrey, "The MSR Cookbook: Mining a Decade of Research," in *Proceedings of the International Conference on Mining Software Repositories (MSR)*, 2013, pp. 343–352.
[33] I. Herraiz, D. M. German, and A. E. Hassan, "On the Distribution of Source Code File Sizes," in *Proceedings of the International Conference on Software Technologies (ICSOFT)*, 2011, pp. 5–14.
[34] J. Jiarpakdee, C. Tantithamthavorn, and A. E. Hassan, "Online Appendix for "The Impact of Correlated Metrics on the Interpretation of Defect Models"," *https://github.com/SAILResearch/collinearity-pitfalls*, 2018.
[35] J. Jiarpakdee, C. Tantithamthavorn, A. Ihara, and K. Matsumoto, "A Study of Redundant Metrics in Defect Prediction Datasets," in *Proceedings of the International Symposium on Software Reliability Engineering Workshops (ISSREW)*, 2016, pp. 51–52.
[36] J. Jiarpakdee, C. Tantithamthavorn, and C. Treude, "Autospearman: Automatically mitigating correlated metrics for interpreting defect models," in *Proceeding of the International Conference on Software Maintenance and Evolution (ICSME)*, 2018, pp. 92–103.
[37] S. Kabinna, W. Shang, C.-P. Bezemer, and A. E. Hassan, "Examining the Stability of Logging Statements," in *Proceedings of the International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2016, pp. 326–337.
[38] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, and N. Ubayashi, "A Large-Scale Empirical Study of Just-In-Time Quality Assurance," *Transactions on Software Engineering (TSE)*, vol. 39, no. 6, pp. 757–773, 2013.
[39] J. Keung, E. Kocaguneli, and T. Menzies, "Finding Conclusion Stability for Selecting the Best Effort Predictor in Software Effort Estimation," *Automated Software Engineering*, vol. 20, no. 4, pp. 543–567, 2013.
[40] F. Khomh, M. Di Penta, and Y.-G. Gueheneuc, "An Exploratory Study of the Impact of Code Smells on Software Change-proneness," in *Proceedings of the Working Conference on Reverse Engineering (WCRE)*, 2009, pp. 75–84.
[41] F. Khomh, M. Di Penta, Y.-G. Guéhéneuc, and G. Antoniol, "An Exploratory Study of the Impact of Antipatterns on Class Change- and Fault-proneness," *Empirical Software Engineering (EMSE)*, vol. 17, no. 3, pp. 243–275, 2012.
[42] S. Kim, T. Zimmermann, E. J. Whitehead Jr, and A. Zeller, "Predicting Faults from Cached History," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2007, pp. 489–498.
[43] S. Kirbas, B. Caglayan, T. Hall, S. Counsell, D. Bowes, A. Sen, and A. Bener, "The Relationship between Evolutionary Coupling and Defects in Large Industrial Software," *Journal of Software: Evolution and Process*, vol. 29, no. 4, 2017.
[44] A. G. Koru and H. Liu, "An Investigation of the Effect of Module Size on Defect Prediction Using Static Measures," *Software Engineering Notes (SEN)*, vol. 30, pp. 1–5, 2005.
[45] H. C. Kraemer, G. A. Morgan, N. L. Leech, J. A. Gliner, J. J. Vaske, and R. J. Harmon, "Measures of Clinical Significance," *Journal of the American Academy of Child & Adolescent Psychiatry (JAACAP)*, vol. 42, no. 12, pp. 1524–1529, 2003.
[46] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," *Transactions on Software Engineering (TSE)*, vol. 34, no. 4, pp. 485–496, 2008.
[47] H. Lu, E. Kocaguneli, and B. Cukic, "Defect Prediction between Software Versions with Active Learning and Dimensionality Reduction," in *Proceedings of the International Symposium on Software Reliability Engineering (ISSRE)*, 2014, pp. 312–322.
[48] B. W. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochimica et Biophysica Acta (BBA)-Protein Structure*, vol. 405, no. 2, pp. 442–451, 1975.
[49] T. J. McCabe, "A Complexity Measure," *Transactions on Software Engineering (TSE)*, no. 4, pp. 308–320, 1976.
[50] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "The Impact of Code Review Coverage and Code Review Participation on Software Quality," in *Proceedings of the International Conference on Mining Software Repositories (MSR)*, 2014, pp. 192–201.
[51] T. Mende, "Replication of Defect Prediction Studies: Problems, Pitfalls and Recommendations," in *Proceedings of the International Conference on Predictive Models in Software Engineering (PROMISE)*, 2010, pp. 1–10.
[52] T. Mende and R. Koschke, "Revisiting the Evaluation of Defect Prediction Models," *Proceedings of the International Conference on Predictive Models in Software Engineering (PROMISE)*, pp. 7–16, 2009.
[53] A. Meneely, L. Williams, W. Snipes, and J. Osborne, "Predicting Failures with Developer Networks and Social Network Analysis," in *Proceedings of the International Symposium on Foundations of Software Engineering (FSE)*, 2008, pp. 13–23.
[54] T. Menzies, J. Greenwald, and A. Frank, "Data Mining Static Code Attributes to Learn Defect Predictors," *Transactions on Software Engineering (TSE)*, vol. 33, no. 1, pp. 2–13, 2007.
[55] A. T. Misirli, E. Shihab, and Y. Kamei, "Studying High Impact Fix-Inducing Changes," *Empirical Software Engineering (EMSE)*, vol. 21, no. 2, pp. 605–641, 2016.
[56] R. Morales, S. McIntosh, and F. Khomh, "Do Code Review Practices Impact Design Quality? : A Case Study of the Qt, VTK, and ITK Projects," in *Proceedings of the International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2015, pp. 171–180.
[57] N. Nagappan and T. Ball, "Use of Relative Code Churn Measures to Predict System Defect Density," *Proceedings of the International Conference on Software Engineering (ICSE)*, pp. 284–292, 2005.
[58] N. Nagappan, T. Ball, and A. Zeller, "Mining Metrics to Predict Component Failures," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2006, pp. 452–461.
[59] N. Nagappan, A. Zeller, T. Zimmermann, K. Herzig, and B. Murphy, "Change Bursts as Defect Predictors," in *Proceedings of the International Symposium on Software Reliability Engineering (ISSRE)*, 2010, pp. 309–318.
[60] J. Nam, W. Fu, S. Kim, T. Menzies, and L. Tan, "Heterogeneous Defect Prediction," *Transactions on Software Engineering (TSE)*, p. In Press, 2017.
[61] F. Rahman and P. Devanbu, "Ownership, experience and defects: a fine-grained study of authorship," in *Proceedings of the International*

Conference on Software Engineering (ICSE), 2011, pp. 491–500.
[62] ——, "How, and Why, Process Metrics are Better," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2013, pp. 432–441.
[63] F. Rahman, D. Posnett, I. Herraiz, and P. Devanbu, "Sample size vs. bias in defect prediction," in *Proceedings of the Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2013, pp. 147–157.
[64] G. K. Rajbahadur, S. Wang, Y. Kamei, and A. E. Hassan, "The Impact of Using Regression Models to Build Defect Classifiers," in *Proceedings of the International Conference on Mining Software Repositories (MSR)*, 2017, pp. 135–145.
[65] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A Large Scale Study of Programming Languages and Code Quality in Github," in *Proceedings of the International Symposium on Foundations of Software Engineering (FSE)*, 2014, pp. 155–165.
[66] M. Shepperd, D. Bowes, and T. Hall, "Researcher Bias: The Use of Machine Learning in Software Defect Prediction," *Transactions on Software Engineering (TSE)*, vol. 40, no. 6, pp. 603–616, 2014.
[67] E. Shihab, "An Exploration of Challenges Limiting Pragmatic Software Defect Prediction," Ph.D. dissertation, Queen's University, 2012.
[68] E. Shihab, C. Bird, and T. Zimmermann, "The Effect of Branching Strategies on Software Quality," in *Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2012, pp. 301–310.
[69] E. Shihab, Z. M. Jiang, W. M. Ibrahim, B. Adams, and A. E. Hassan, "Understanding the Impact of Code and Process Metrics on Post-release Defects: A Case Study on the Eclipse Project," in *Proceedings of the International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2010, pp. 4–10.
[70] J. Shimagaki, Y. Kamei, S. McIntosh, A. E. Hassan, and N. Ubayashi, "A Study of the Quality-Impacting Practices of Modern Code Review at Sony Mobile," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2016, pp. 212–221.
[71] Y. Shin, A. Meneely, L. Williams, and J. A. Osborne, "Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities," *Transactions on Software Engineering (TSE)*, vol. 37, no. 6, pp. 772–787, 2011.
[72] C. Tantithamthavorn, "Towards a Better Understanding of the Impact of Experimental Components on Defect Prediction Modelling," in *Companion Proceeding of the International Conference on Software Engineering (ICSE)*, 2016, p. 867870.
[73] ——, "ScottKnottESD : The Scott-Knott Effect Size Difference (ESD) Test. R package version 2.0," *Software available at URL: https://cran.r-project.org/web/packages/ScottKnottESD*, 2017.
[74] C. Tantithamthavorn and A. E. Hassan, "An Experience Report on Defect Modelling in Practice: Pitfalls and Challenges," in *In Proceedings of the International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP)*, 2018, pp. 286–295.
[75] C. Tantithamthavorn, A. E. Hassan, and K. Matsumoto, "The Impact of Class Rebalancing Techniques on The Performance and Interpretation of Defect Prediction Models," *Transactions on Software Engineering (TSE)*, 2018.
[76] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, "Automated Parameter Optimization of Classification Techniques for Defect Prediction Models," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2016, pp. 321–332.
[77] ——, "Comments on "Researcher Bias: The Use of Machine Learning in Software Defect Prediction"," *Transactions on Software Engineering (TSE)*, vol. 42, no. 11, pp. 1092–1094, 2016.
[78] ——, "An Empirical Comparison of Model Validation Techniques for Defect Prediction Models," *Transactions on Software Engineering (TSE)*, vol. 43, no. 1, pp. 1–18, 2017.
[79] ——, "The Impact of Automated Parameter Optimization on Defect Prediction Models," *Transactions on Software Engineering (TSE)*, p. In Press, 2018.
[80] R. C. Team and contributors worldwide, "stats : The R Stats Package. R Package. Version 3.4.0," 2017.
[81] P. Thongtanunam, S. McIntosh, A. E. Hassan, and H. Iida, "Revisiting Code Ownership and its Relationship with Software Quality in the Scope of Modern Code Review," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2016, pp. 1039–1050.
[82] ——, "Review Participation in Modern Code Review," *Empirical Software Engineering (EMSE)*, vol. 22, no. 2, pp. 768–817, 2017.
[83] Y. Tian, M. Nagappan, D. Lo, and A. E. Hassan, "What Are the Characteristics of High-Rated Apps? A Case Study on Free Android Applications," in *Proceedings of the International Conference on Software Maintenance and Evolution (ICSME)*, 2015, pp. 301–310.
[84] S. Yathish, J. Jiarpakdee, P. Thongtanunam, and C. Tantithamthavorn, "Mining Software Defects: Should We Consider Affected Releases?" in *In Proceedings of the International Conference on Software Engineering (ICSE)*, 2019, p. To Appear.
[85] F. Zhang, A. E. Hassan, S. McIntosh, and Y. Zou, "The Use of Summation to Aggregate Software Metrics Hinders the Performance of Defect Prediction Models," *Transactions on Software Engineering (TSE)*, vol. 43, no. 5, pp. 476–491, 2017.
[86] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "Cross-project Defect Prediction," in *Proceedings of the Joint Meeting of the European Software Engineering Conference and the Symposium on the Foundations of Software Engineering (ESEC/FSE)*, 2009, pp. 91–100.
[87] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting Defects for Eclipse," in *Proceedings of the International Workshop on Predictor Models in Software Engineering (PROMISE)*, 2007, pp. 9–19.
[88] T. Zimmermann, A. Zeller, P. Weissgerber, and S. Diehl, "Mining Version Histories to Guide Software Changes," *Transactions on Software Engineering (TSE)*, vol. 31, no. 6, pp. 429–445, 2005.



Jirayus Jiarpakdee received the B.E. degree from Kasetsart University, Thailand, and the M.E. degree from NAIST, Japan. He is currently a Ph.D. candidate at Monash University, Australia. His research interests include empirical software engineering and mining software repositories (MSR). The goal of his Ph.D. is to apply the knowledge of statistical modelling, experimental design, and software engineering in order to tackle experimental issues that have an impact on the interpretation of defect prediction models.

Chakkrit Tantithamthavorn is a lecturer at the Faculty of Information Technology, Monash University, Australia. His work has been published at several top-tier software engineering venues (e.g., TSE, ICSE, EMSE). His research interests include empirical software engineering and mining software repositories (MSR). He received the B.E. degree from Kasetsart University, Thailand, the M.E. and Ph.D. degrees from NAIST, Japan. More about Chakkrit and his work is available online at http://chakkrit.com.

Ahmed E. Hassan is the Canada Research Chair (CRC) in Software Analytics, and the NSERC/BlackBerry Software Engineering Chair at the School of Computing at Queen's University, Canada. He received a PhD in Computer Science from the University of Waterloo. He spearheaded the creation of the Mining Software Repositories (MSR) conference and its research community. More about Ahmed and his work is available online at http://sail.cs.queensu.ca/.